Analyses and plots for Chapter 17 – Discussion

Constituency and convergence in the Americas

Sandra Auderset

16 July, 2024

Contents

This script provides all the code for the analyses and figures presented in the discussion chapter (17) in the order of the chapter text.

Section 3

Map of the sample languages

Create map of the sample languages. This includes all the languages which are discussed in the chapters of the volume, Chacobo, which was discussed in Tallman (2021), and Siksika, which could not be included in the first volume but is fully analyzed (Natalie Weber, p.c.). The map is created using Stadia Maps (stadiamaps.com) with ggmaps Kahle & Wickham (2013).

```
# read in constituency database
domains <- read_tsv(here("domains.tsv")) %>%
  modify_if(is.character, as.factor)
```

```
## Rows: 463 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (11): Language_Name, Language_ID, Domain_ID, Domain_Type, Abstract_Type,...
## dbl (10): Serial_Order, Left_Edge, Right_Edge, Convergence, Relative_Converg...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# read in metadata file
metadata <- read_tsv(here("input/metadata.tsv"))
```

```
## Rows: 37 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (7): Language_Name, Alternative_Name, Short_Name, Language_ID, Glottocod...
## dbl (3): No, Latitude, Longitude
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# subset for languages of volume 1 and Chacobo and Siksika
metadata_sub <- metadata %>%
  filter(Contribution == "Vol1" | Language_ID == "siks1238" | Language_ID == "chac1251")

# find min/max values of lat and long for map
summary(metadata_sub$Latitude)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -28.10  -12.13   14.64    9.60   18.10   60.31
```

```r
summary(metadata_sub$Longitude)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -161.49  -97.46  -71.00  -83.71  -66.17  -51.67
```

```r
# get basemap
americas <- get_stadiamap(bbox = c(left = -167, bottom = -36, right = -40, top = 64),
maptype = "stamen_terrain", zoom = 5)
```

```
## i © Stadia Maps © Stamen Design © OpenMapTiles © OpenStreetMap contributors.
## i 144 tiles needed, this may take a while (try a smaller zoom?)
```

```r
# add points for languages
sample_map <- ggmap(americas) +
  geom_point(aes(x = Longitude, y = Latitude), data = metadata_sub, alpha = 1, size = 3)
  +
  geom_label_repel(aes(x = Longitude, y = Latitude, label = Short_Name), data =
  metadata_sub, box.padding = unit(.3, "lines"), label.padding = unit(.2, "lines"),
  max.overlaps = 30, size = 3) +
  theme_map()
sample_map
```

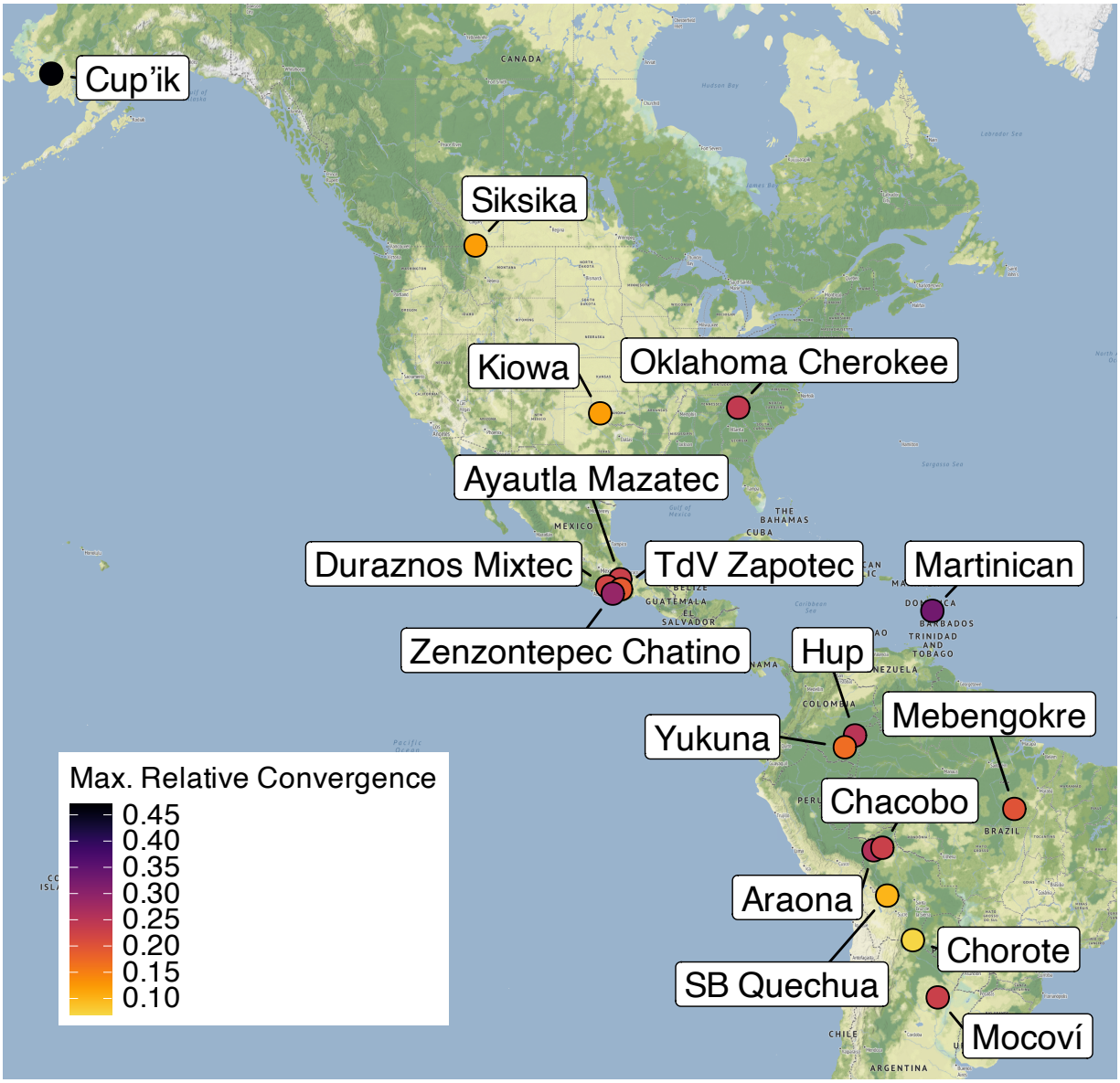
We color the points by maximum number of relative convergences per language.

```
# merge with domains file
metadata_sub_conv <- domains %>%
  select(Language_ID, Relative_Convergence) %>%
  group_by(Language_ID) %>%
  slice(which.max(Relative_Convergence)) %>%
  left_join(., metadata_sub)
```

```
## Joining with `by = join_by(Language_ID)`
```

```
# update map
sample_map_relconv <- ggmap(americas) +
  geom_point(aes(x = Longitude, y = Latitude, fill = Relative_Convergence), data =
  metadata_sub_conv, size = 4, pch = 21) +
  scale_fill_viridis(
    option = "inferno", direction = -1, end = 0.9,
    name = "Max. Relative Convergence",
    breaks = seq(0, 0.45, by = 0.05)
  ) +
  geom_label_repel(aes(x = Longitude, y = Latitude, label = Short_Name), data =
  metadata_sub, box.padding = .6, point.padding = 1, max.overlaps = 30,
  min.segment.length = 0.1, size = 5) +
  theme_map() +
  theme(
    legend.position = c(0.05, 0.05),
    legend.direction = "vertical",
    legend.title = element_text(size = 12),
```

```
    legend.text = element_text(size = 12)
  )
sample_map_relconv
```



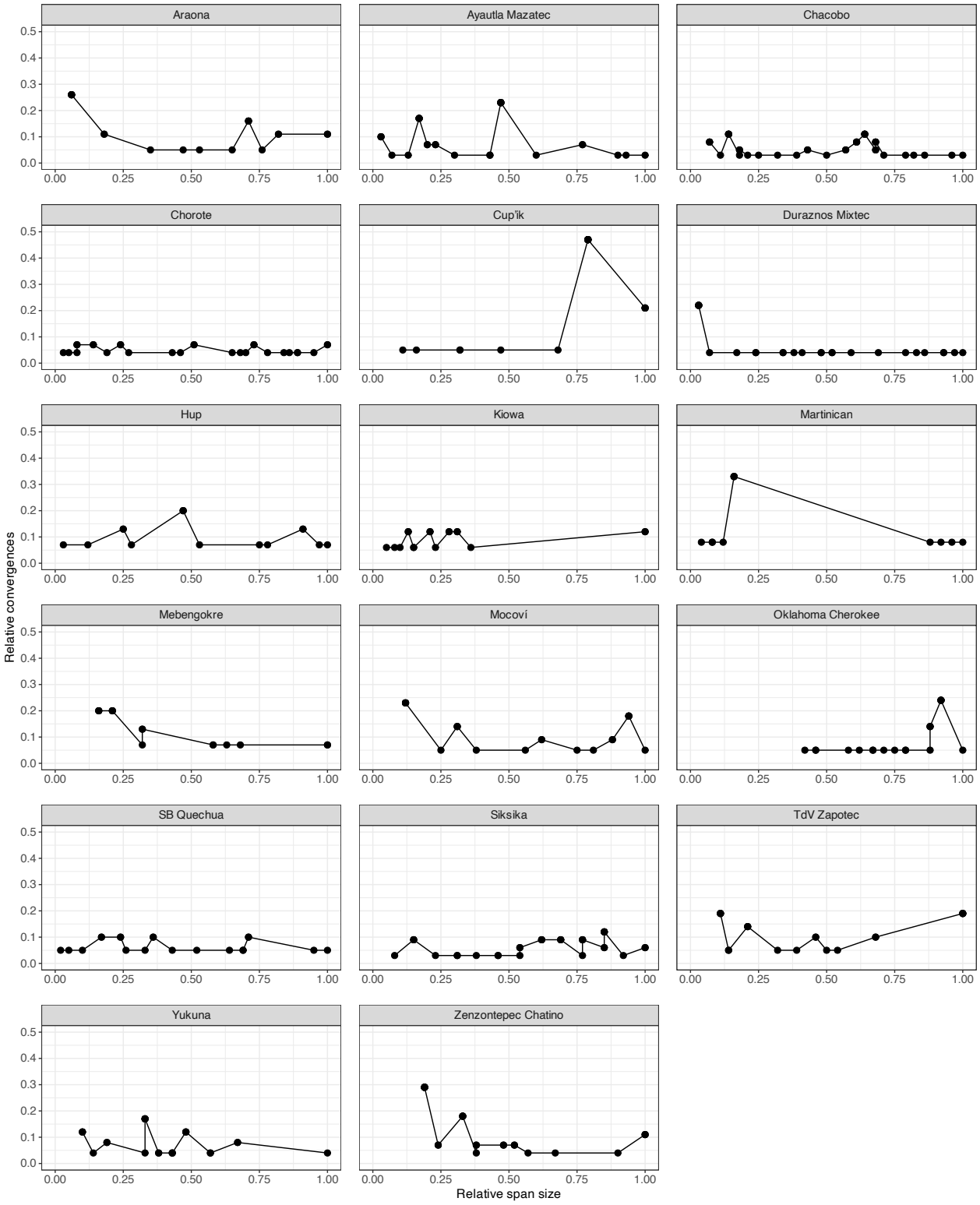
Dot plot of convergences and span size

We plot the relative convergences against the relative span size per sample language. We use only verbal domains as we currently have too little data for comparing nominal domains as well.

```
# exclude all domains that are not verbal; add short name of languages for plotting
domains_verbal <- domains %>%
  filter(str_detect(Planar_ID, "verbal")) %>%
  filter(!is.na(Convergence)) %>%
  left_join(., select(metadata_sub, Language_ID, Short_Name)) %>%
  arrange(Relative_Size, Relative_Convergence)
```

```
## Joining with `by = join_by(Language_ID)`
```

```
# make dot plot per language
plot_rconv_facet <- ggplot(aes(x = Relative_Size, y = Relative_Convergence, group =
Language_Name), data = domains_verbal) +
  geom_point(aes(), size = 2.5) +
  geom_line(linewidth = 0.5) +
  facet_wrap(~Short_Name, nrow = 6, ncol = 3, scales = "free_x") +
  ylab("Relative convergences") +
  xlab("Relative span size") +
  scale_x_continuous(limits = c(0, 1), breaks = seq(0, 1, 0.25)) +
  scale_y_continuous(limits = c(0, 0.5)) +
  theme_bw() +
  theme(
    axis.text = element_text(size = 11),
    axis.title = element_text(size = 12),
    strip.text.x = element_text(size = 11),
    panel.spacing = unit(1.2, "lines")
  )
plot_rconv_facet
```



Density plot of span size by type

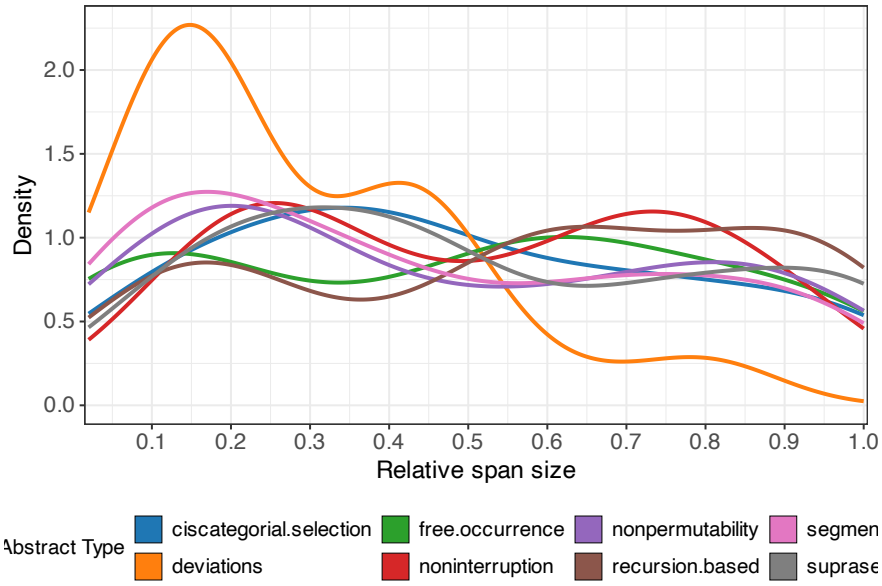
We plot densities of relative span size by abstract type of test domain. Since we are doing this across languages, we exclude abstract types that have less than 10 data points, as these are not informative.

```r
# list all abstract types
levels(domains$Abstract_Type)
```

```
##  [1] "ciscategorial.selection" "deviations"
##  [3] "free.occurrence"         "idiom"
##  [5] "noninterruption"         "nonpermutability"
##  [7] "pausing"                 "play.language"
##  [9] "proform"                 "recursion.based"
## [11] "repair"                  "segmental"
## [13] "suprasegmental"
```

```r
# remove types with under 10 data points
include_at <- domains_verbal %>%
  count(Abstract_Type) %>%
  arrange(desc(n)) %>%
  filter(n > 5) %>%
  droplevels() %>%
  pull(Abstract_Type)
domains_verbal_sub <- domains_verbal %>%
  filter(Abstract_Type %in% include_at) %>%
  droplevels()

# make density plot
plot_density_type <- ggplot(aes(x = Relative_Size, group = Abstract_Type), data =
domains_verbal_sub) +
  geom_density(aes(fill = Abstract_Type, color = Abstract_Type), alpha = 0, linewidth =
  1) +
  scale_color_d3(palette = "category10", guide = "none") +
  scale_fill_d3(palette = "category10", name = "Abstract Type") +
  guides(fill = guide_legend(override.aes = list(alpha = 1, linewidth = 0))) +
  labs(x = "Relative span size", y = "Density", ) +
  scale_x_continuous(expand = c(0.005, 0), breaks = seq(0, 1, 0.1)) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    legend.key.size = unit(1, "lines"),
    legend.text = element_text(size = 11),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 13)
  )
plot_density_type
```



Section 6

Subsection 6.2

Correlation plots across abstract types

To look at correlations between abstract types of tests, we recode all variables into binary numeric ones, where 0 indicates absence and 1 presence. We then sum these binary values per abstract type of test for each language and span. We repeat the same process, but for each language and left and right edges, respectively. We create these plots with the 'ggcorrplot' package Kassambara (2023).

```
# recode the variables to numeric
abstract_type <- domains_verbal_sub %>%
  pivot_wider(., names_from = Abstract_Type, values_from = Abstract_Type) %>%
  mutate(across(nonpermutability:noninterruption, ~ if_else(is.na(.x), 0, 1)))
# sum overlaps by language and abstract type - for the whole span
abstract_type_span <- abstract_type %>%
  group_by(Language_ID, Left_Edge, Right_Edge) %>%
  summarise(
    Nonpermut. = sum(nonpermutability),
    Noninterrupt. = sum(noninterruption),
    Selection. = sum(ciscategorial.selection),
    Segmental. = sum(segmental),
    Free_occur. = sum(free.occurrence),
    Recursion. = sum(recursion.based),
    Supraseg. = sum(suprasegmental),
    Deviations. = sum(deviations)
  ) %>%
  ungroup() %>%
  select(-c(Language_ID, Left_Edge, Right_Edge))
```

```
## `summarise()` has grouped output by 'Language_ID', 'Left_Edge'. You can
## override using the `.groups` argument.
```

```
glimpse(abstract_type_span)
```

```
## Rows: 236
## Columns: 8
## $ Nonpermut.    <dbl> 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,~
## $ Noninterrupt. <dbl> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,~
## $ Selection.    <dbl> 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ Segmental.    <dbl> 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,~
## $ Free_occur.   <dbl> 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Recursion.    <dbl> 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 2, 0,~
## $ Supraseg.     <dbl> 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,~
## $ Deviations.   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
```

```
# # sum overlaps by language and abstract type - for the right edge
abstract_type_right <- abstract_type %>%
  group_by(Language_ID, Right_Edge) %>%
  summarise(
    Nonpermut. = sum(nonpermutability),
    Noninterrupt. = sum(noninterruption),
    Selection. = sum(ciscategorial.selection),
    Segmental. = sum(segmental),
    Free_occur. = sum(free.occurrence),
    Recursion. = sum(recursion.based),
    Supraseg. = sum(suprasegmental),
    Deviations. = sum(deviations)
  ) %>%
  ungroup() %>%
  select(-c(Language_ID, Right_Edge))
```

```
## `summarise()` has grouped output by 'Language_ID'. You can override using the
## `.groups` argument.
```

```
glimpse(abstract_type_right)
```

```
## Rows: 139
## Columns: 8
## $ Nonpermut.    <dbl> 2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,~
## $ Noninterrupt. <dbl> 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1,~
## $ Selection.    <dbl> 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2,~
## $ Segmental.    <dbl> 2, 0, 0, 1, 1, 0, 0, 2, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 2,~
## $ Free_occur.   <dbl> 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,~
## $ Recursion.    <dbl> 1, 0, 0, 1, 1, 0, 1, 2, 1, 0, 0, 1, 2, 0, 1, 1, 1, 0, 1,~
## $ Supraseg.     <dbl> 0, 1, 0, 0, 0, 0, 1, 2, 2, 2, 0, 2, 0, 1, 0, 0, 0, 0, 1,~
## $ Deviations.   <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,~
```

```
# sum overlaps by language and abstract type - for the left edge
abstract_type_left <- abstract_type %>%
  group_by(Language_ID, Left_Edge) %>%
  summarise(
    Nonpermut. = sum(nonpermutability),
    Noninterrupt. = sum(noninterruption),
    Selection. = sum(ciscategorial.selection),
    Segmental. = sum(segmental),
    Free_occur. = sum(free.occurrence),
    Recursion. = sum(recursion.based),
    Supraseg. = sum(suprasegmental),
    Deviations. = sum(deviations)
```

```
  ) %>%
  ungroup() %>%
  select(-c(Language_ID, Left_Edge))
```

```
## `summarise()` has grouped output by 'Language_ID'. You can override using the
## `.groups` argument.
```

```
glimpse(abstract_type_left)
```

```
## Rows: 96
## Columns: 8
## $ Nonpermut.    <dbl> 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 2, 0, 0, 0, 0, 3, 0, 0, 2,~
## $ Noninterrupt. <dbl> 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 1,~
## $ Selection.    <dbl> 1, 1, 1, 0, 0, 0, 0, 2, 0, 0, 1, 1, 0, 1, 0, 0, 2, 0, 1,~
## $ Segmental.    <dbl> 0, 2, 2, 0, 0, 0, 1, 0, 2, 1, 2, 0, 0, 0, 1, 6, 1, 0, 3,~
## $ Free_occur.   <dbl> 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 2, 0, 0, 0, 0, 1, 1, 0, 1,~
## $ Recursion.    <dbl> 1, 2, 1, 1, 2, 1, 0, 2, 0, 0, 4, 0, 0, 0, 5, 7, 0, 2, 2,~
## $ Supraseg.     <dbl> 0, 1, 1, 1, 0, 0, 0, 6, 0, 2, 2, 0, 0, 2, 1, 3, 1, 0, 0,~
## $ Deviations.   <dbl> 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,~
```

We then calculate Kendall's tau for each of these data sets and plot the results as correlation plots.

```
# make correlation matrices
corr_span <- cor(abstract_type_span, method = "kendall")
corr_left <- cor(abstract_type_left, method = "kendall")
corr_right <- cor(abstract_type_right, method = "kendall")

# plot correlations
plot_corr_span <- ggcorrplot(corr_span,
  type = "lower",
  lab = TRUE,
  lab_size = 4
)
plot_corr_span
```

```
plot_corr_left <- ggcorrplot(corr_left,
  type = "lower",
  lab = TRUE,
  lab_size = 5,
  tl.cex = 12
)

plot_corr_right <- ggcorrplot(corr_right,
  type = "lower",
  lab = TRUE,
  lab_size = 5,
  tl.cex = 12
)
```

Correlation tables for cross-language and minimal/maximal fractures

We also want to look at more fine-grained fractures. We thus create new subsets with the cross-language fractures and the minimal/maximal fractures, where available. For ciscategorial selection, free occurrence, recursion based, segmental, and suprasegmental test domains, we use the minimal/maximal fractures. For deviations from biuniqueness, non-permutability, and non-interruptability we use the cross-language fractures.

```
# data set for comparing cross-language fractures and min-max domains
cross_l <- domains_verbal_sub %>%
  mutate(MinMax_Fracture = case_match(
    MinMax_Fracture,
    "minimal" ~ "min",
    "maximal" ~ "max"
  )) %>%
  mutate(abstract_cross_l = case_when(
    Abstract_Type == "deviations" ~ paste(Abstract_Type, CrossL_Fracture, sep = "_"),
    Abstract_Type == "nonpermutability" ~ paste(Abstract_Type, CrossL_Fracture, sep =
"_"),
    Abstract_Type == "noninterruption" ~ paste(Abstract_Type, CrossL_Fracture, sep =
"_"),
    Abstract_Type == "ciscategorial.selection" ~ paste(Abstract_Type, MinMax_Fracture,
sep = "_"),
    Abstract_Type == "free.occurrence" ~ paste(Abstract_Type, MinMax_Fracture, sep =
"_"),
    Abstract_Type == "recursion.based" ~ paste(Abstract_Type, MinMax_Fracture, sep =
"_"),
    Abstract_Type == "segmental" ~ paste(Abstract_Type, MinMax_Fracture, sep = "_"),
    Abstract_Type == "suprasegmental" ~ paste(Abstract_Type, MinMax_Fracture, sep = "_")
  )) %>%
  mutate(
    abstract_cross_l = str_remove_all(abstract_cross_l, "_NA"),
    abstract_cross_l = if_else(Abstract_Type == "deviations", Abstract_Type,
    abstract_cross_l)
  ) %>%
  arrange(abstract_cross_l) %>%
  pivot_wider(., names_from = abstract_cross_l, values_from = abstract_cross_l) %>%
  mutate(across(ciscategorial.selection:suprasegmental_min, ~ if_else(is.na(.x), 0, 1)))
```

We look at correlations between minimal and maximal domains and cross-language fractures, respectively. We do this first for the whole span and then for the right and left edge. Since there are many test pairs and we are only

interested in correlations that are at least moderate, we summarize the results in tables omitting values between -0.2 and 0.2.

```
# make data set for correlations using whole span
cross_l_span <- cross_l %>%
  group_by(Language_ID, Left_Edge, Right_Edge) %>%
  summarise(
    Noninterrupt_simpl = sum(noninterruption_simplex),
    Noninterrupt_compl = sum(noninterruption_complex),
    Noninterrupt_multipos = sum(noninterruption_multipositional),
    Nonpermut_rigid = sum(nonpermutability_rigid),
    Nonpermut_scopal = sum(nonpermutability_scopal),
    Selection_min = sum(ciscategorial.selection_min),
    Selection_max = sum(ciscategorial.selection_max),
    Recursion_min = sum(recursion.based_min),
    Recursion_max = sum(recursion.based_max),
    FreeOccur_min = sum(free.occurrence_min),
    FreeOccur_max = sum(free.occurrence_max),
    Deviations = sum(deviations),
    Segmental_min = sum(segmental_min),
    Segmental_max = sum(segmental_max),
    Supraseg_min = sum(suprasegmental_min),
    Supraseg_max = sum(suprasegmental_min)
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'Language_ID', 'Left_Edge'. You can
## override using the `.groups` argument.
```

```
glimpse(cross_l_span)
```

```
## Rows: 236
## Columns: 19
## $ Language_ID           <chr> "arao1248", "arao1248", "arao1248", "arao1248", ~
## $ Left_Edge             <dbl> 1, 4, 4, 4, 4, 4, 6, 6, 6, 6, 2, 2, 3, 3, 6, 13,~
## $ Right_Edge            <dbl> 17, 6, 14, 15, 16, 17, 6, 11, 13, 14, 29, 31, 20~
## $ Noninterrupt_simpl    <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Noninterrupt_compl    <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ~
## $ Noninterrupt_multipos <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Nonpermut_rigid       <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Nonpermut_scopal      <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ~
## $ Selection_min         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ~
## $ Selection_max         <dbl> 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Recursion_min         <dbl> 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, ~
## $ Recursion_max         <dbl> 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, ~
## $ FreeOccur_min         <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ FreeOccur_max         <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Deviations            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Segmental_min         <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Segmental_max         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ~
## $ Supraseg_min          <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Supraseg_max          <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ~
```

```
# minimal domains and cross-language fractures
cross_l_min <- select(cross_l_span, -ends_with("_max"), -c(Language_ID:Right_Edge))
```

```r
# maximal domains and cross-language fractures
cross_l_max <- select(cross_l_span, -ends_with("_min"), -c(Language_ID:Right_Edge))
# correlation matrices
corr_cross_l_min <- cor(cross_l_min, method = "kendall")
corr_cross_l_max <- cor(cross_l_max, method = "kendall")
```

```r
# same but with edge convergences instead of whole span
# left edge:
cross_l_left <- cross_l %>%
  group_by(Language_ID, Left_Edge) %>%
  summarise(
    Noninterrupt_simpl = sum(noninterruption_simplex),
    Noninterrupt_compl = sum(noninterruption_complex),
    Noninterrupt_multipos = sum(noninterruption_multipositional),
    Nonpermut_rigid = sum(nonpermutability_rigid),
    Nonpermut_scopal = sum(nonpermutability_scopal),
    Selection_min = sum(ciscategorial.selection_min),
    Selection_max = sum(ciscategorial.selection_max),
    Recursion_min = sum(recursion.based_min),
    Recursion_max = sum(recursion.based_max),
    FreeOccur_min = sum(free.occurrence_min),
    FreeOccur_max = sum(free.occurrence_max),
    Deviations = sum(deviations),
    Segmental_min = sum(segmental_min),
    Segmental_max = sum(segmental_max),
    Supraseg_min = sum(suprasegmental_min),
    Supraseg_max = sum(suprasegmental_min)
  ) %>%
  ungroup()

# right edge:
cross_l_right <- cross_l %>%
  group_by(Language_ID, Right_Edge) %>%
  summarise(
    Noninterrupt_simpl = sum(noninterruption_simplex),
    Noninterrupt_compl = sum(noninterruption_complex),
    Noninterrupt_multipos = sum(noninterruption_multipositional),
    Nonpermut_rigid = sum(nonpermutability_rigid),
    Nonpermut_scopal = sum(nonpermutability_scopal),
    Selection_min = sum(ciscategorial.selection_min),
    Selection_max = sum(ciscategorial.selection_max),
    Recursion_min = sum(recursion.based_min),
    Recursion_max = sum(recursion.based_max),
    FreeOccur_min = sum(free.occurrence_min),
    FreeOccur_max = sum(free.occurrence_max),
    Deviations = sum(deviations),
    Segmental_min = sum(segmental_min),
    Segmental_max = sum(segmental_max),
    Supraseg_min = sum(suprasegmental_min),
    Supraseg_max = sum(suprasegmental_min)
  ) %>%
  ungroup()
```

```
# minimal domains and cross-language fractures - left edges
cross_l_min_left <- select(cross_l_left, -ends_with("_max"), -c(Language_ID:Left_Edge))
# maximal domains and cross-language fractures
cross_l_max_left <- select(cross_l_left, -ends_with("_min"), -c(Language_ID:Left_Edge))

# minimal domains and cross-language fractures - right edges
cross_l_min_right <- select(cross_l_right, -ends_with("_max"),
-c(Language_ID:Right_Edge))
# maximal domains and cross-language fractures
cross_l_max_right <- select(cross_l_right, -ends_with("_min"),
-c(Language_ID:Right_Edge))
```

To create the correlation tables efficiently, we set up a custom function.

```
# set up function to create correlation dataframes
# for span
corr_to_df1 <- function(x, y, z) {
  df1 <- cor(x, method = "kendall") %>%
    as.data.frame() %>%
    rownames_to_column(., var = "Test1") %>%
    pivot_longer(-Test1, names_to = "Test2", values_to = paste0(y, ".", z))
  # full correlation df
  corr_full <- df1 %>%
    distinct(across(starts_with("M")), .keep_all = TRUE) %>%
    filter(Test1 != Test2) %>%
    kable(
      booktabs = TRUE, linesep = "",
      align = "llrr"
    )
  # excluding weak correlations and highlighting cells with higher correlations;
  exporting table to latex
  corr_higher <- df1 %>%
    distinct(across(starts_with("M")), .keep_all = TRUE) %>%
    filter(Test1 != Test2) %>%
    mutate(across(starts_with("M"), ~ round(.x, 2))) %>%
    mutate(across(starts_with("M"), ~ case_when(
      .x > 0.3 ~ paste0("\\cellcolor{red!45}", .x),
      between(.x, 0.2, 0.3) ~ paste0("\\cellcolor{red!25}", .x),
      .x < -0.3 ~ paste0("\\cellcolor{blue!45}", .x),
      between(.x, 0.2, 0.3) ~ paste0("\\cellcolor{blue!25}", .x),
      .default = paste(.x)
    ))) %>%
    mutate(across(starts_with("Test"), ~ str_replace_all(.x, "_", "."))) %>%
    filter(if_any(starts_with("M"), ~ str_detect(., "cellcolor"))) %>%
    kable(
      booktabs = TRUE, format = "latex", escape = FALSE, linesep = "",
      align = "llrr"
    ) %>%
    kable_styling()
  # return full df and latex table
  return(list(corr_full, corr_higher))
}

# for edges
```

```r
corr_to_df2 <- function(x, y, z, a, b, c) {
  df1 <- cor(x, method = "kendall") %>%
    as.data.frame() %>%
    rownames_to_column(., var = "Test1") %>%
    pivot_longer(-Test1, names_to = "Test2", values_to = paste0(y, ".", z))
  df2 <- cor(a, method = "kendall") %>%
    as.data.frame() %>%
    rownames_to_column(., var = "Test1") %>%
    pivot_longer(-Test1, names_to = "Test2", values_to = paste0(b, ".", c)) %>%
    select(-c(Test1, Test2))
  # full correlation df
  corr_full <- bind_cols(df1, df2) %>%
    distinct(across(starts_with("M")), .keep_all = TRUE) %>%
    filter(Test1 != Test2) %>%
    kable(
      booktabs = TRUE, linesep = "",
      align = "llrr"
    )
  # excluding weak correlations and highlighting cells with higher correlations;
  exporting table to latex
  corr_higher <- bind_cols(df1, df2) %>%
    distinct(across(starts_with("M")), .keep_all = TRUE) %>%
    filter(Test1 != Test2) %>%
    mutate(across(starts_with("M"), ~ round(.x, 2))) %>%
    mutate(across(starts_with("M"), ~ case_when(
      .x > 0.3 ~ paste0("\\cellcolor{red!45}", .x),
      between(.x, 0.2, 0.3) ~ paste0("\\cellcolor{red!25}", .x),
      .x < -0.3 ~ paste0("\\cellcolor{blue!45}", .x),
      between(.x, 0.2, 0.3) ~ paste0("\\cellcolor{blue!25}", .x),
      .default = paste(.x)
    ))) %>%
    mutate(across(starts_with("Test"), ~ str_replace_all(.x, "_", "."))) %>%
    filter(if_any(starts_with("M"), ~ str_detect(., "cellcolor"))) %>%
    kable(
      booktabs = TRUE, format = "latex", escape = FALSE, linesep = "",
      align = "llrr"
    ) %>%
    kable_styling()
  # return full df and latex table
  return(list(corr_full, corr_higher))
}
```

We apply the function to each subset and display the full correlation tables. The highlighted tables are exported for inclusion in the chapter.

```r
# cross-language + min/max - whole span
corr_cross_l_min_span <- corr_to_df1(cross_l_min, "Min", "Span")
corr_cross_l_min_span[[1]]
```

Test1	Test2	Min.Span
Noninterrupt_simpl	Noninterrupt_compl	-0.0350877
Noninterrupt_simpl	Noninterrupt_multipos	-0.0245959
Noninterrupt_simpl	Nonpermut_rigid	0.1431491

Test1	Test2	Min.Span
Noninterrupt_simpl	Nonpermut_scopal	-0.0327498
Noninterrupt_simpl	Selection_min	-0.0470398
Noninterrupt_simpl	Recursion_min	0.0910499
Noninterrupt_simpl	FreeOccur_min	0.0306309
Noninterrupt_simpl	Deviations	0.0024339
Noninterrupt_simpl	Segmental_min	0.0860252
Noninterrupt_simpl	Supraseg_min	0.0085362
Noninterrupt_compl	Nonpermut_rigid	-0.0488008
Noninterrupt_compl	Recursion_min	0.0140833
Noninterrupt_compl	FreeOccur_min	-0.0554274
Noninterrupt_compl	Deviations	-0.0681498
Noninterrupt_compl	Segmental_min	0.0109136
Noninterrupt_compl	Supraseg_min	0.0828325
Noninterrupt_multipos	Nonpermut_rigid	-0.0342086
Noninterrupt_multipos	Nonpermut_scopal	-0.0229571
Noninterrupt_multipos	Selection_min	0.2449508
Noninterrupt_multipos	Recursion_min	-0.0440803
Noninterrupt_multipos	FreeOccur_min	-0.0388537
Noninterrupt_multipos	Deviations	-0.0477720
Noninterrupt_multipos	Segmental_min	-0.0450017
Noninterrupt_multipos	Supraseg_min	-0.0460968
Nonpermut_rigid	Nonpermut_scopal	-0.0455492
Nonpermut_rigid	Selection_min	0.0081001
Nonpermut_rigid	Recursion_min	0.0267238
Nonpermut_rigid	FreeOccur_min	0.0505817
Nonpermut_rigid	Deviations	-0.0947844
Nonpermut_rigid	Segmental_min	0.0280959
Nonpermut_rigid	Supraseg_min	-0.0363498
Nonpermut_scopal	Selection_min	-0.0439055
Nonpermut_scopal	Recursion_min	-0.0586934
Nonpermut_scopal	FreeOccur_min	-0.0517343
Nonpermut_scopal	Deviations	-0.0636090
Nonpermut_scopal	Segmental_min	-0.0599204
Nonpermut_scopal	Supraseg_min	-0.0613785
Selection_min	Recursion_min	0.0336211
Selection_min	FreeOccur_min	0.1234739
Selection_min	Deviations	0.0230740
Selection_min	Segmental_min	0.0290165
Selection_min	Supraseg_min	0.0825896
Recursion_min	FreeOccur_min	0.2590134
Recursion_min	Deviations	0.0087240
Recursion_min	Segmental_min	0.1131364
Recursion_min	Supraseg_min	-0.0736586
FreeOccur_min	Deviations	0.0910615
FreeOccur_min	Segmental_min	0.2982587
FreeOccur_min	Supraseg_min	0.0937864
Deviations	Segmental_min	0.0488071
Deviations	Supraseg_min	0.0391242
Segmental_min	Supraseg_min	0.1863994

```
corr_cross_l_max_span <- corr_to_df1(cross_l_max, "Max", "Span")
corr_cross_l_max_span[[1]]
```

Test1	Test2	Max.Span
Noninterrupt_simpl	Noninterrupt_compl	-0.0350877
Noninterrupt_simpl	Noninterrupt_multipos	-0.0245959
Noninterrupt_simpl	Nonpermut_rigid	0.1431491
Noninterrupt_simpl	Nonpermut_scopal	-0.0327498
Noninterrupt_simpl	Selection_max	0.0383744
Noninterrupt_simpl	Recursion_max	-0.0681671
Noninterrupt_simpl	FreeOccur_max	0.2027476
Noninterrupt_simpl	Deviations	0.0024339
Noninterrupt_simpl	Segmental_max	-0.0596055
Noninterrupt_simpl	Supraseg_max	0.0085362
Noninterrupt_compl	Nonpermut_rigid	-0.0488008
Noninterrupt_compl	Selection_max	-0.0521891
Noninterrupt_compl	FreeOccur_max	-0.0554274
Noninterrupt_compl	Deviations	-0.0681498
Noninterrupt_compl	Supraseg_max	0.0828325
Noninterrupt_multipos	Nonpermut_rigid	-0.0342086
Noninterrupt_multipos	Nonpermut_scopal	-0.0229571
Noninterrupt_multipos	Selection_max	-0.0365838
Noninterrupt_multipos	Recursion_max	-0.0477841
Noninterrupt_multipos	FreeOccur_max	0.0817974
Noninterrupt_multipos	Deviations	-0.0477720
Noninterrupt_multipos	Segmental_max	-0.0417825
Noninterrupt_multipos	Supraseg_max	-0.0460968
Nonpermut_rigid	Nonpermut_scopal	-0.0455492
Nonpermut_rigid	Selection_max	-0.0054084
Nonpermut_rigid	Recursion_max	-0.0948084
Nonpermut_rigid	FreeOccur_max	0.0505817
Nonpermut_rigid	Deviations	-0.0947844
Nonpermut_rigid	Segmental_max	-0.0829007
Nonpermut_rigid	Supraseg_max	-0.0363498
Nonpermut_scopal	Selection_max	-0.0487117
Nonpermut_scopal	Recursion_max	-0.0636251
Nonpermut_scopal	FreeOccur_max	0.1318640
Nonpermut_scopal	Deviations	-0.0636090
Nonpermut_scopal	Segmental_max	-0.0556340
Nonpermut_scopal	Supraseg_max	-0.0613785
Selection_max	Recursion_max	0.0468614
Selection_max	FreeOccur_max	0.0380306
Selection_max	Deviations	-0.0519604
Selection_max	Segmental_max	-0.0886566
Selection_max	Supraseg_max	0.0061962
Recursion_max	FreeOccur_max	-0.0137639
Recursion_max	Deviations	-0.1323989
Recursion_max	Segmental_max	-0.0259420
Recursion_max	Supraseg_max	-0.0872182
FreeOccur_max	Deviations	-0.0607077
FreeOccur_max	Segmental_max	-0.0418979
FreeOccur_max	Supraseg_max	-0.0050468

Test1	Test2	Max.Span
Deviations	Segmental_max	-0.1157700
Deviations	Supraseg_max	0.0391242
Segmental_max	Supraseg_max	-0.0164050

```
# cross-language + minimal - left/right
corr_cross_l_min <- corr_to_df2(cross_l_min_left, "Min", "Left", cross_l_min_right,
"Min", "Right")
corr_cross_l_min[[1]]
```

Test1	Test2	Min.Left	Min.Right
Noninterrupt_simpl	Noninterrupt_compl	0.0586075	0.2041985
Noninterrupt_simpl	Noninterrupt_multipos	0.1538093	-0.0425375
Noninterrupt_simpl	Nonpermut_rigid	0.3817291	0.1131670
Noninterrupt_simpl	Nonpermut_scopal	0.0734833	0.0843454
Noninterrupt_simpl	Selection_min	0.2571236	-0.0827024
Noninterrupt_simpl	Recursion_min	0.0884481	0.2159912
Noninterrupt_simpl	FreeOccur_min	-0.0396129	-0.0039560
Noninterrupt_simpl	Deviations	0.2272164	0.0413788
Noninterrupt_simpl	Segmental_min	0.2919756	0.0507138
Noninterrupt_simpl	Supraseg_min	0.1680648	-0.0320786
Noninterrupt_compl	Noninterrupt_multipos	-0.0628695	0.1422349
Noninterrupt_compl	Nonpermut_rigid	0.1194933	-0.0859497
Noninterrupt_compl	Nonpermut_scopal	0.2053565	-0.0569077
Noninterrupt_compl	Selection_min	0.0981295	-0.0827024
Noninterrupt_compl	Recursion_min	0.4573045	0.0534044
Noninterrupt_compl	FreeOccur_min	0.1340079	-0.0039560
Noninterrupt_compl	Deviations	0.0763835	0.0540223
Noninterrupt_compl	Segmental_min	0.2138178	0.0642767
Noninterrupt_compl	Supraseg_min	0.2277638	0.1277312
Noninterrupt_multipos	Nonpermut_rigid	0.0729153	0.0788268
Noninterrupt_multipos	Nonpermut_scopal	0.1420172	0.1571412
Noninterrupt_multipos	Selection_min	0.3882387	0.2283690
Noninterrupt_multipos	Recursion_min	-0.0947373	0.0371990
Noninterrupt_multipos	FreeOccur_min	-0.1035780	0.0606219
Noninterrupt_multipos	Deviations	0.0236155	0.0464363
Noninterrupt_multipos	Segmental_min	0.0442955	0.0542194
Noninterrupt_multipos	Supraseg_min	0.0158837	0.0333135
Nonpermut_rigid	Nonpermut_scopal	0.2577841	0.2379919
Nonpermut_rigid	Selection_min	0.1995048	0.1147347
Nonpermut_rigid	Recursion_min	0.1786094	0.0904180
Nonpermut_rigid	FreeOccur_min	0.0414455	0.1484730
Nonpermut_rigid	Deviations	0.0140816	0.1419272
Nonpermut_rigid	Segmental_min	0.1165267	-0.0230173
Nonpermut_rigid	Supraseg_min	0.3614843	0.0223279
Nonpermut_scopal	Selection_min	0.1200822	0.0322425
Nonpermut_scopal	Recursion_min	0.0910146	-0.0151667
Nonpermut_scopal	FreeOccur_min	-0.0387557	0.0084261
Nonpermut_scopal	Deviations	-0.0640026	-0.1028252
Nonpermut_scopal	Segmental_min	0.0650839	-0.0163285
Nonpermut_scopal	Supraseg_min	0.1719241	-0.1043535

Test1	Test2	Min.Left	Min.Right
Selection_min	Recursion_min	0.1302715	0.0408683
Selection_min	FreeOccur_min	0.2556631	0.3643023
Selection_min	Deviations	0.1775783	0.0982878
Selection_min	Segmental_min	0.1876479	0.0378763
Selection_min	Supraseg_min	0.2406169	0.1060675
Recursion_min	FreeOccur_min	0.3116368	0.2927496
Recursion_min	Deviations	0.1809762	0.0636153
Recursion_min	Segmental_min	0.3039147	0.1155138
Recursion_min	Supraseg_min	0.2436265	-0.0043524
FreeOccur_min	Deviations	0.2231660	0.2946544
FreeOccur_min	Segmental_min	0.4292868	0.4054280
FreeOccur_min	Supraseg_min	0.3697661	0.2289099
Deviations	Segmental_min	0.3843538	0.1275054
Deviations	Supraseg_min	0.2306674	0.1510526
Segmental_min	Supraseg_min	0.3617206	0.2278026

```
# cross-language + maximal - left/right
corr_cross_l_max <- corr_to_df2(cross_l_max_left, "Max", "Left", cross_l_max_right,
"Max", "Right")
corr_cross_l_max[[1]]
```

Test1	Test2	Max.Left	Max.Right
Noninterrupt_simpl	Noninterrupt_compl	0.0586075	0.2041985
Noninterrupt_simpl	Noninterrupt_multipos	0.1538093	-0.0425375
Noninterrupt_simpl	Nonpermut_rigid	0.3817291	0.1131670
Noninterrupt_simpl	Nonpermut_scopal	0.0734833	0.0843454
Noninterrupt_simpl	Selection_max	0.1882274	0.0020349
Noninterrupt_simpl	Recursion_max	-0.0520398	0.0467851
Noninterrupt_simpl	FreeOccur_max	0.2574835	0.2774557
Noninterrupt_simpl	Deviations	0.2272164	0.0413788
Noninterrupt_simpl	Segmental_max	0.2722157	-0.0079399
Noninterrupt_simpl	Supraseg_max	0.1680648	-0.0320786
Noninterrupt_compl	Noninterrupt_multipos	-0.0628695	0.1422349
Noninterrupt_compl	Nonpermut_rigid	0.1194933	-0.0859497
Noninterrupt_compl	Nonpermut_scopal	0.2053565	-0.0569077
Noninterrupt_compl	Selection_max	0.0575922	0.0020349
Noninterrupt_compl	Recursion_max	0.1677723	-0.1076057
Noninterrupt_compl	FreeOccur_max	0.2286017	0.0929342
Noninterrupt_compl	Deviations	0.0763835	0.0540223
Noninterrupt_compl	Segmental_max	0.1244123	-0.0079399
Noninterrupt_compl	Supraseg_max	0.2277638	0.1277312
Noninterrupt_multipos	Nonpermut_rigid	0.0729153	0.0788268
Noninterrupt_multipos	Nonpermut_scopal	0.1420172	0.1571412
Noninterrupt_multipos	Selection_max	0.1763842	-0.0642551
Noninterrupt_multipos	Recursion_max	-0.1166327	-0.0749530
Noninterrupt_multipos	FreeOccur_max	0.0272574	0.0647335
Noninterrupt_multipos	Deviations	0.0236155	0.0464363
Noninterrupt_multipos	Segmental_max	0.1642564	-0.0714714
Noninterrupt_multipos	Supraseg_max	0.0158837	0.0333135
Nonpermut_rigid	Nonpermut_scopal	0.2577841	0.2379919

Test1	Test2	Max.Left	Max.Right
Nonpermut_rigid	Selection_max	0.0737923	0.0117103
Nonpermut_rigid	Recursion_max	-0.0565064	-0.0355571
Nonpermut_rigid	FreeOccur_max	0.5241642	0.0096045
Nonpermut_rigid	Deviations	0.0140816	0.1419272
Nonpermut_rigid	Segmental_max	0.1826726	-0.0733522
Nonpermut_rigid	Supraseg_max	0.3614843	0.0223279
Nonpermut_scopal	Selection_max	-0.0251447	-0.0859620
Nonpermut_scopal	Recursion_max	0.0326811	-0.1002739
Nonpermut_scopal	FreeOccur_max	0.2629103	0.1111662
Nonpermut_scopal	Deviations	-0.0640026	-0.1028252
Nonpermut_scopal	Segmental_max	0.1522914	-0.0956161
Nonpermut_scopal	Supraseg_max	0.1719241	-0.1043535
Selection_max	Recursion_max	0.0839477	0.0773226
Selection_max	FreeOccur_max	0.2489805	0.1321240
Selection_max	Deviations	0.0845809	0.1147964
Selection_max	Segmental_max	-0.0293431	0.0468888
Selection_max	Supraseg_max	0.0464228	0.1716444
Recursion_max	FreeOccur_max	0.0243731	0.0160968
Recursion_max	Deviations	0.0755752	-0.0961583
Recursion_max	Segmental_max	0.0442958	0.0524054
Recursion_max	Supraseg_max	0.0970535	-0.0428956
FreeOccur_max	Deviations	-0.0311684	-0.0073009
FreeOccur_max	Segmental_max	0.0614584	0.1146591
FreeOccur_max	Supraseg_max	0.2781544	-0.0498077
Deviations	Segmental_max	0.1460861	-0.0838708
Deviations	Supraseg_max	0.2306674	0.1510526
Segmental_max	Supraseg_max	0.2952724	-0.0235193

Subsection 6.3

Density plot of relative convergences across span and edges

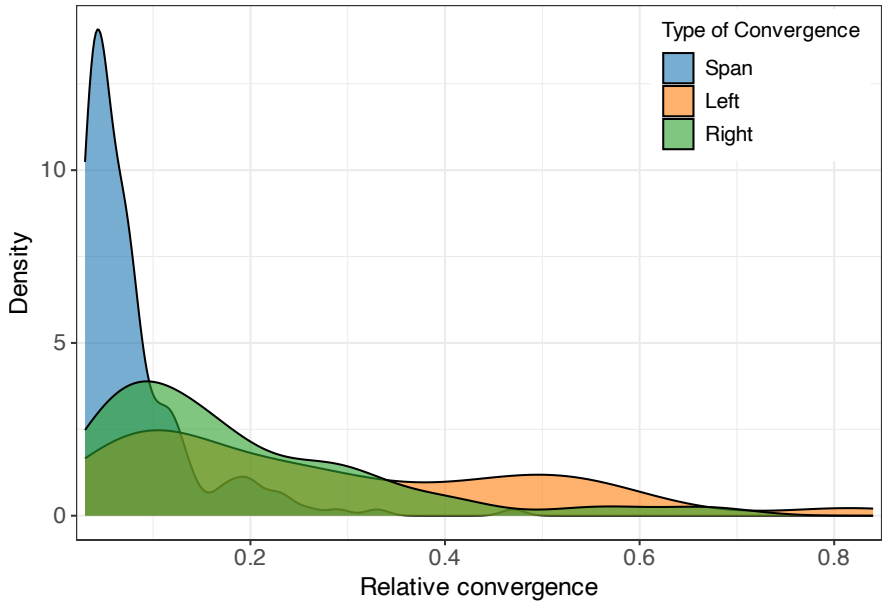
In order to assess if there are overall tendencies for some domains to converge than others, we explore the density distributions of span and edge convergences pooled across languages of the sample. We do this by calculating the the convergences across the span, and on the right and left edge and displaying the densities of these distributions.

```
# calculate right and left edge convergences
domains_verbal <- domains_verbal %>%
  group_by(Language_ID, Left_Edge) %>%
  mutate(Convergence_Left = n()) %>%
  group_by(Language_ID, Right_Edge) %>%
  mutate(Convergence_Right = n()) %>%
  ungroup() %>%
  mutate(Relative_Convergence_Left = round(Convergence_Left / Tests_Total, 2),
  Relative_Convergence_Right = round(Convergence_Right / Tests_Total, 2))

# data set for plotting density
domains_verbal_density <- domains_verbal %>%
  rename(
    Span = Relative_Convergence,
    Left = Relative_Convergence_Left,
    Right = Relative_Convergence_Right
  ) %>%
```

```
  distinct(Language_ID, Span, Right, Left) %>%
  pivot_longer(-Language_ID, names_to = "Convergence_Type", values_to =
  "Relative_Convergence") %>%
  mutate(Convergence_Type = factor(Convergence_Type, levels = c("Span", "Left",
  "Right")))

# density plot across all languages
plot_density_all <- ggplot(data = domains_verbal_density, aes(x = Relative_Convergence,
group = Convergence_Type, fill = Convergence_Type)) +
  geom_density(alpha = 0.6) +
  scale_fill_d3(name = "Type of Convergence") +
  # guides(fill = guide_legend(override.aes = list(alpha = 1, linewidth = 0))) +
  labs(x = "Relative convergence", y = "Density", ) +
  scale_x_continuous(expand = c(0.005, 0.005), breaks = seq(0, 1, 0.2)) +
  theme_bw() +
  theme(
    legend.position = c(0.85, 0.85),
    # legend.key.size = unit(1, "lines"),
    legend.text = element_text(size = 11),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 13),
    strip.text.x = element_text(size = 11)
  )
plot_density_all
```

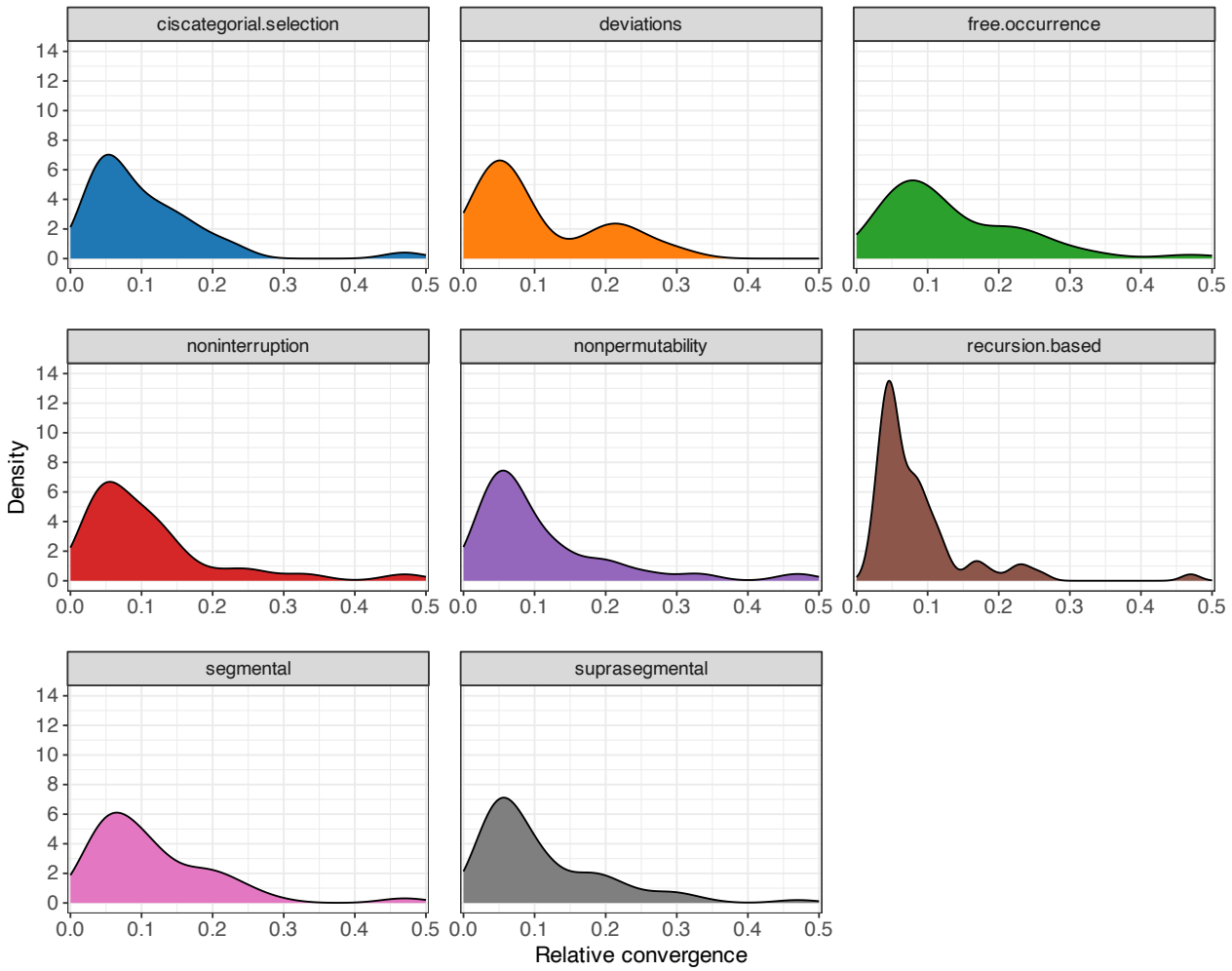


Density plots of relative span convergence per abstract type

We do the same, but separating the tests by abstract type. We exclude types with fewer than 5 data points, because these are not informative.

```
# facet density plot of relative convergence per abstract type
plot_dens_conv_atype <- ggplot(aes(x = Relative_Convergence, group = Abstract_Type), data
= domains_verbal_sub) +
  geom_density(aes(fill = Abstract_Type)) +
```

```
  scale_fill_d3(guide = "none") +
  facet_wrap(~Abstract_Type, scales = "free_x") +
  labs(x = "Relative convergence", y = "Density", ) +
  scale_x_continuous(expand = c(0.008, 0), breaks = seq(0, 5, 0.1), limits = c(0, 0.5)) +
  scale_y_continuous(breaks = seq(0, 14, 2), limits = c(0, 14)) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    legend.key.size = unit(1, "lines"),
    legend.text = element_text(size = 11),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 13),
    strip.text.x = element_text(size = 11),
    panel.spacing = unit(1.3, "lines"),
    strip.clip = "off"
  )
plot_dens_conv_atype
```



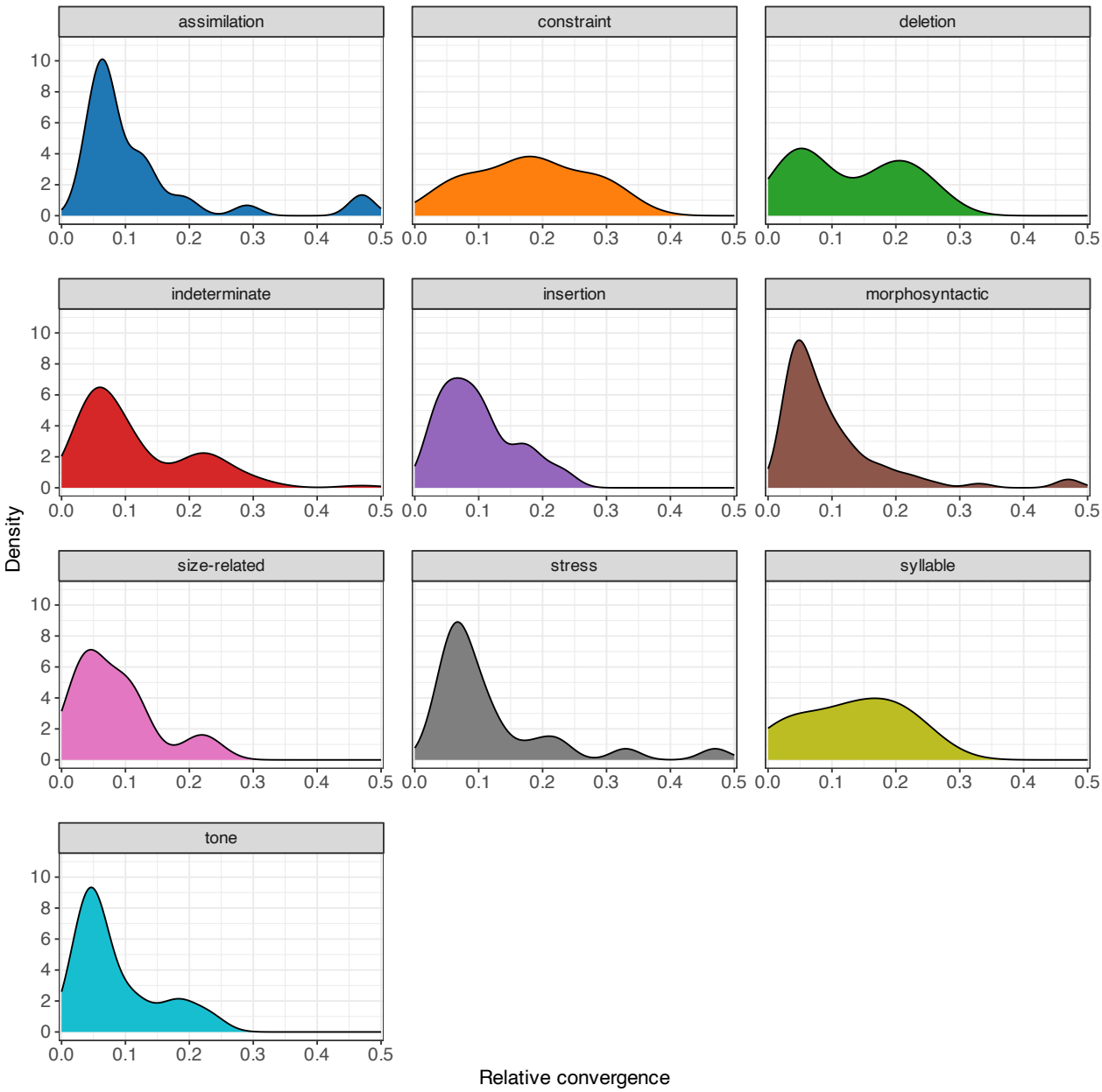
Density plots of relative span convergence per prosodic word domain

Next, we look at the distributions by type of prosodic word domain. We exclude types with fewer than 5 data points, because these are not informative.

```
# exclude categories with few data points
keep_pw <- domains_verbal_sub %>%
  count(PW_Pattern) %>%
  filter(n > 5) %>%
  droplevels() %>%
  pull(PW_Pattern)

domains_verbal_sub_pw <- domains_verbal_sub %>%
  filter(PW_Pattern %in% keep_pw) %>%
  droplevels()

# make plot
plot_dens_conv_pw <- ggplot(aes(x = Relative_Convergence, group = PW_Pattern), data =
domains_verbal_sub_pw) +
  geom_density(aes(fill = PW_Pattern)) +
  scale_fill_d3(guide = "none") +
  facet_wrap(~PW_Pattern, scales = "free_x", ncol = 3) +
  labs(x = "Relative convergence", y = "Density", ) +
  scale_x_continuous(expand = c(0.008, 0), breaks = seq(0, 5, 0.1), limits = c(0, 0.5)) +
  scale_y_continuous(breaks = seq(0, 11, 2), limits = c(0, 11)) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    legend.key.size = unit(1, "lines"),
    legend.text = element_text(size = 11),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 13),
    strip.text.x = element_text(size = 11),
    panel.spacing = unit(1.3, "lines"),
    strip.clip = "off"
  )
plot_dens_conv_pw
```

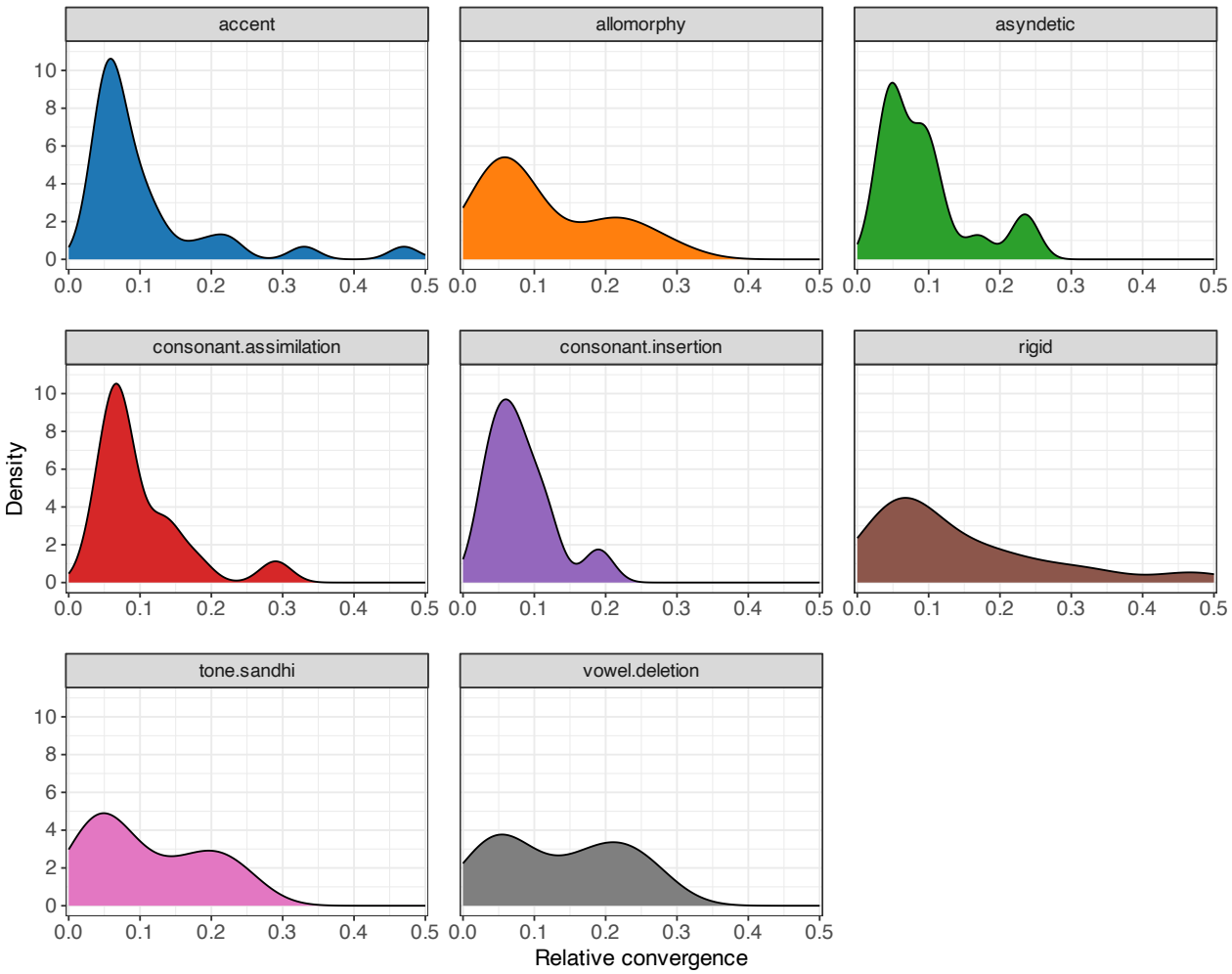


Density plots of relative span convergence across cross-language fractures

Lastly, we do the same but look at cross-linguistic fractures with more than 10 tokens each.

```
# take subset with fractures that have more than 10 tokens; exclude NA
keep_clf <- domains_verbal %>%
  filter(!is.na(CrossL_Fracture)) %>%
  count(CrossL_Fracture) %>%
  filter(n > 10) %>%
  droplevels() %>%
  pull(CrossL_Fracture)
domains_verbal_clf <- domains_verbal %>%
  filter(CrossL_Fracture %in% keep_clf) %>%
  droplevels()
```

```r
# make plot
plot_dens_conv_clf <- ggplot(aes(x = Relative_Convergence, group = CrossL_Fracture), data
= domains_verbal_clf) +
  geom_density(aes(fill = CrossL_Fracture)) +
  scale_fill_d3(guide = "none") +
  facet_wrap(~CrossL_Fracture, scales = "free_x", ncol = 3) +
  labs(x = "Relative convergence", y = "Density", ) +
  scale_x_continuous(expand = c(0.008, 0), breaks = seq(0, 5, 0.1), limits = c(0, 0.5)) +
  scale_y_continuous(breaks = seq(0, 11, 2), limits = c(0, 11)) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    legend.key.size = unit(1, "lines"),
    legend.text = element_text(size = 11),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 13),
    strip.text.x = element_text(size = 11),
    panel.spacing = unit(1.3, "lines"),
    strip.clip = "off"
  )
plot_dens_conv_clf
```



Random Forest analysis

To check whether certain domains predict convergence levels, we apply a random forest model to the data. We compare the output of the model to the baseline to calculate the accuracy. The model barely outperforms the baseline, which means it does not really predict anything. Therefore the variable importance is not meaningful. We include the plot for completeness.

```
# function for calculating baseline and accuracy
rf_acc <- function(x) {
  cm <- as.matrix(x$confusion[, -3])
  bl <- round(max(rowSums(cm)) / sum(cm), 4)
  acc <- round(sum(diag(cm)) / sum(cm), 4)
  diff <- round(acc - bl, 4)
  returnlist <- c("baseline" = bl, "accuracy" = acc, "difference" = diff)
}
# set seed for reproducibility
set.seed(200)

# subset for relevant variables, turn into factors
pw_patterns <- domains_verbal_sub %>%
  mutate(
    PW_Pattern = str_replace_all(PW_Pattern, "-", "."),
```

```
    PW_Pattern = str_replace_all(PW_Pattern, " ", ".")
  ) %>%
  pivot_wider(., names_from = PW_Pattern, values_from = PW_Pattern, names_prefix = "pw_")
  %>%
  mutate(across(starts_with("pw_"), ~ if_else(is.na(.x), 0, 1))) %>%
  select(Serial_Order, Language_ID, pw_morphosyntactic:pw_other.process)

# add together with other fracture data sets; subset for applying random forest
all_fractures <- abstract_type %>%
  select(Language_ID:noninterruption) %>%
  bind_cols(select(cross_l, -any_of(colnames(abstract_type)), -Abstract_Type)) %>%
  bind_cols(select(pw_patterns, pw_morphosyntactic:pw_other.process)) %>%
  select(Convergence, nonpermutability:pw_other.process) %>%
  mutate(Convergence = as.factor(Convergence))
glimpse(all_fractures)
```

```
## Rows: 381
## Columns: 37
## $ Convergence                    <fct> 1, 1, 1, 3, 3, 3, 6, 6, 6, 6, 6, 6, 1,~
## $ nonpermutability               <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ free.occurrence                <dbl> 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,~
## $ recursion.based                <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,~
## $ segmental                      <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ suprasegmental                 <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
## $ ciscategorial.selection        <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,~
## $ deviations                     <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,~
## $ noninterruption                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ ciscategorial.selection_max    <dbl> 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ ciscategorial.selection_min    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ free.occurrence_max            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ free.occurrence_min            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ noninterruption_complex        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ noninterruption_multipositional <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ noninterruption_simplex        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ nonpermutability_rigid         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ nonpermutability_scopal        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ recursion.based_max            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ recursion.based_min            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ segmental_max                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ segmental_min                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ suprasegmental_max             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ suprasegmental_min             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_morphosyntactic             <dbl> 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,~
## $ pw_indeterminate               <dbl> 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1,~
## $ pw_dissimilation               <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_tone                        <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_size.related                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ pw_stress                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_assimilation                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_deletion                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_insertion                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_syllable                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_allomorphy                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

```
## $ pw_constraint                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pw_other.process               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

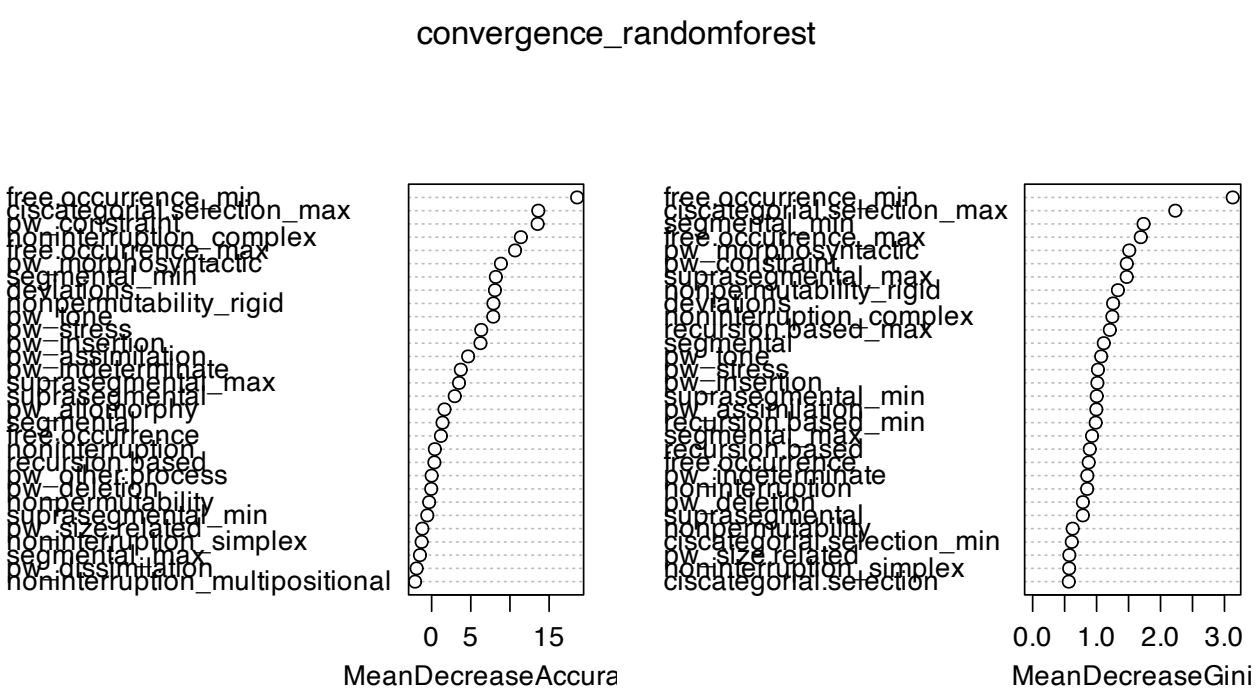
```
# apply random forest algorithm
convergence_randomforest <- randomForest(Convergence ~ ., data = all_fractures, ntree =
1000, mtry = 2, importance = TRUE)
convergence_randomforest
```

```
##
## Call:
##  randomForest(formula = Convergence ~ ., data = all_fractures,      ntree = 1000, mtry = 2, importanc
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 58.27%
## Confusion matrix:
##     1 2 3 4 5 6 7 8 9 class.error
## 1 159 0 0 0 0 0 0 0 0           0
## 2  81 0 0 0 0 0 0 0 0           1
## 3  54 0 0 0 0 0 0 0 0           1
## 4  35 0 0 0 0 0 0 0 0           1
## 5  25 0 0 0 0 0 0 0 0           1
## 6   6 0 0 0 0 0 0 0 0           1
## 7   6 0 0 0 0 0 0 0 0           1
## 8   5 2 0 0 0 0 0 0 0           1
## 9   8 0 0 0 0 0 0 0 0           1
```

```
# calculate baseline and accuracy of the model
rf_measures <- rf_acc(convergence_randomforest)
rf_measures
```

```
##   baseline   accuracy difference
##     0.4087     0.4113     0.0026
```

```
# plot variable importance
varImpPlot(convergence_randomforest)
```



Section 7

Convergence and span size in phonological domains

To assess the word bisection thesis, i.e. the idea that there are phonological and grammatical words, we have a closer look at convergence and span sizes in phonological vs. morphosyntactic domains per language. We first create the dataset by counting convergences per language with relative span sizes.

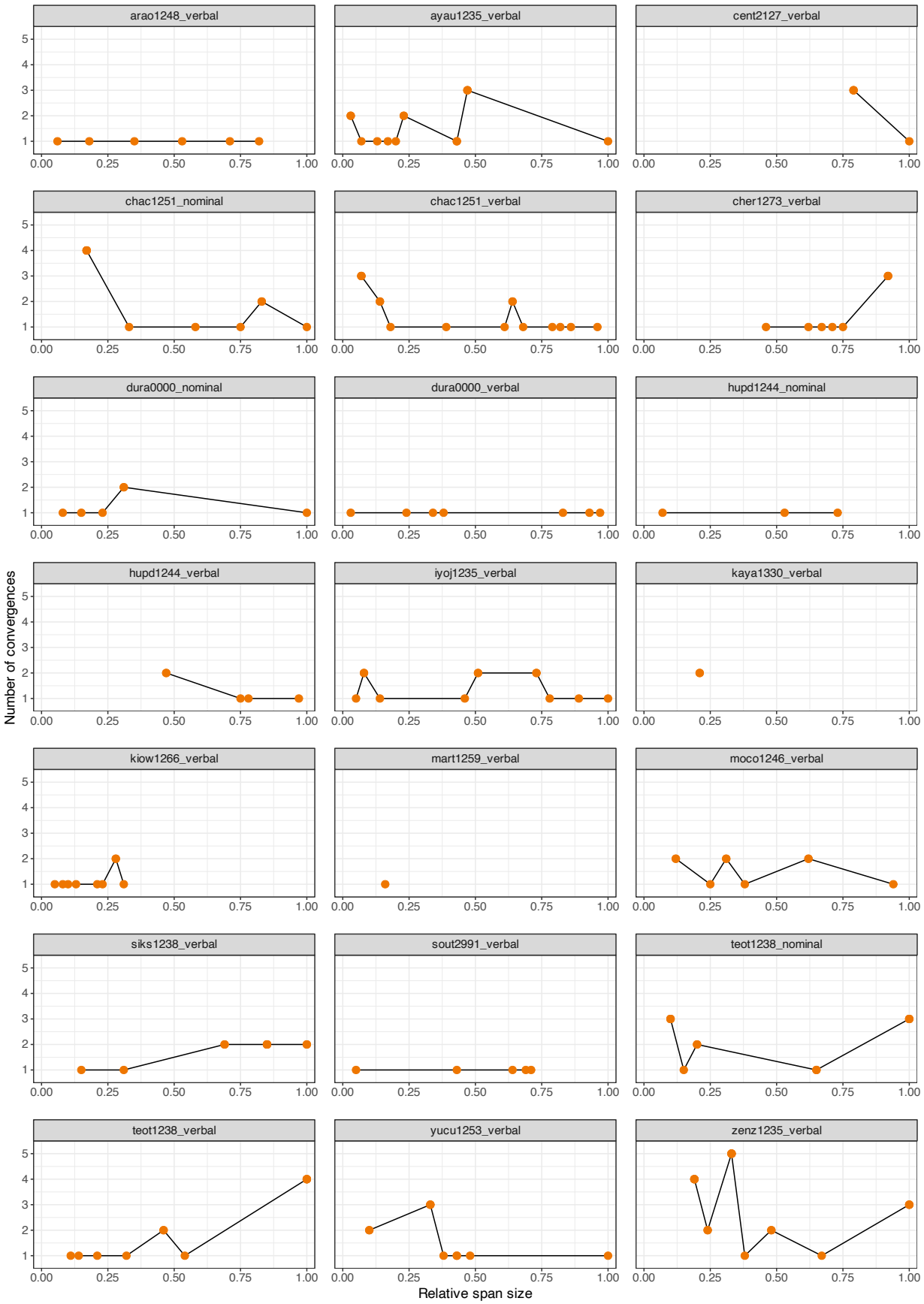
```
# count convergences between those domains per language
conv_bisection <- domains %>%
  unite("Spans", Left_Edge:Right_Edge, sep = "-") %>%
  group_by(Planar_ID, Domain_Type) %>%
  mutate(Total_Tests_D = n()) %>%
  group_by(Planar_ID, Domain_Type, Spans) %>%
  mutate(DConvergence = n()) %>%
  ungroup()

# with indeterminate = morphosyntactic
conv_bisection_lumped <- domains %>%
  unite("Spans", Left_Edge:Right_Edge, sep = "-") %>%
  mutate(Domain_Type_Lumped = if_else(Domain_Type == "indeterminate", "morphosyntactic",
  Domain_Type)) %>%
  group_by(Planar_ID, Domain_Type) %>%
  mutate(Total_Tests_DL = n()) %>%
  group_by(Planar_ID, Domain_Type, Spans) %>%
  mutate(DLConvergence = n()) %>%
  ungroup()

# add back to df for plotting
domains_bisection <- conv_bisection %>%
  left_join(., conv_bisection_lumped) %>%
  mutate(Relative_DConvergence = round(DConvergence / Total_Tests_D, 2),
  Relative_DLConvergence = round(DLConvergence / Total_Tests_DL, 2))
```

We then plot the result for phonological domains.

```
# plot lumped, phonological domains
plot_points_phon <- ggplot(data = filter(domains_bisection, Domain_Type_Lumped ==
"phonological"), aes(x = Relative_Size, y = DLConvergence)) +
  geom_line(linewidth = 0.5) +
  geom_point(size = 3, color = "darkorange2") +
  facet_wrap(~Planar_ID, scales = "free_x", ncol = 3) +
  labs(x = "Relative span size", y = "Number of convergences") +
  scale_x_continuous(expand = c(0, 0.03), breaks = seq(0, 1, 0.25), limits = c(0, 1)) +
  scale_y_continuous(expand = c(0.1, 0.1)) +
  theme_bw() +
  theme(
    legend.position = "top",
    legend.key.size = unit(1, "lines"),
    legend.text = element_text(size = 11),
    axis.text = element_text(size = 11),
    axis.title = element_text(size = 13),
    strip.text.x = element_text(size = 11),
    panel.spacing = unit(1.3, "lines"),
    strip.clip = "off"
  )
plot_points_phon
```


arao1248_verbal
ayau1235_verbal
cent2127_verbal
chac1251_nominal
chac1251_verbal
cher1273_verbal
dura0000_nominal
dura0000_verbal
hupd1244_nominal
hupd1244_verbal
iyoj1235_verbal
kaya1330_verbal
kiow1266_verbal
mart1259_verbal
moco1246_verbal
siks1238_verbal
sout2991_verbal
teot1238_nominal
teot1238_verbal
yucu1253_verbal
zenz1235_verbal
Number of convergences
Relative span size
0.00
0.25
0.50
0.75
1.00
1
2
3
4
5

Convergence and span size in morphosyntactic domains

And the same for morphosyntactic domains.

```
# plot lumped, morphosyntactic domains
plot_points_ms <- ggplot(data = filter(domains_bisection, Domain_Type_Lumped ==
"morphosyntactic"), aes(x = Relative_Size, y = DLConvergence)) +
  geom_line(linewidth = 0.5) +
  geom_point(size = 3, color = "dodgerblue3") +
  facet_wrap(~Planar_ID, scales = "free_x", ncol = 3) +
  labs(x = "Relative span size", y = "Number of convergences") +
  scale_x_continuous(expand = c(0, 0.04), breaks = seq(0, 1, 0.25), limits = c(0, 1)) +
  scale_y_continuous(expand = c(0, 0.3)) +
  theme_bw() +
  theme(
    legend.position = "top",
    legend.key.size = unit(1, "lines"),
    legend.text = element_text(size = 11),
    axis.text = element_text(size = 11),
    axis.title = element_text(size = 13),
    strip.text.x = element_text(size = 11),
    panel.spacing = unit(1.3, "lines"),
    strip.clip = "off"
  )
plot_points_ms
```


arao1248_verbal
ayau1235_verbal
cent2127_verbal
chac1251_nominal
chac1251_verbal
cher1273_verbal
dura0000_nominal
dura0000_verbal
hupd1244_nominal
hupd1244_verbal
iyoj1235_verbal
kaya1330_verbal
kiow1266_verbal
mart1259_verbal
moco1246_verbal
siks1238_verbal
sout2991_verbal
teot1238_nominal
teot1238_verbal
yucu1253_verbal
zenz1235_verbal
Number of convergences
Relative span size
1
2
3
4
0.00
0.25
0.50
0.75
1.00

References

Kahle, David & Wickham, Hadley. 2013. Ggmap: Spatial visualization with ggplot2. *The R Journal*. https://journal.r-project.org/archive/2013-1/kahle-wickham.pdf 5(1). 144–161.

Kassambara, Alboukadel. 2023. Ggcorrplot: Visualization of a correlation matrix using 'ggplot2'. https://CRAN.R-project.org/package=ggcorrplot.

Tallman, Adam JR. 2021. Constituency and coincidence in chácobo (pano). *Studies in Language* 45(2). 321–383.